Complexity of determining exact tolerances for min-max combinatorial optimization problems

Ramaswamy, R.; Chakravarti, N.; Ghosh, D.

Complexity of Determining Exact Tolerances for Min-Max Combinatorial Optimization Problems

Ramkumar Ramaswamy*
Nilotpal Chakravarti†
Diptesh Ghosh‡

SOM-theme A Primary Processes within Firms

Abstract

Suppose that we are given an instance of a combinatorial optimization problem with min-max objective along with an optimal solution for it. Let the cost of a single element be varied. We refer to the range of values of the element's cost for which the given optimal solution remains optimal as its exact tolerance. In this paper we examine the problem of determining the exact tolerance of each element in combinatorial optimization problems with min-max objectives. We show that under very weak assumptions, the exact tolerance of each element can be determined in polynomial time if and only if the original optimization problem can be solved in polynomial time.

Keywords: Sensitivity analysis, exact tolerance, combinatorial optimization, min-max problems, polynomial solvability

Abbreviated title: Exact Tolerances for Min-Max Problems

---

* Infosys Technologies Ltd., 3rd Cross, Electronics City, Hosur Road, Bangalore 561 229, India

† Singapore Airlines, Singapore

‡ Faculty of Economic Sciences, University of Groningen, P.O.Box 800, 9700 AV Groningen, The Netherlands

1. Introduction

Sensitivity analysis of combinatorial optimization problems is a study of the effect of changes in problem data on an optimal solution to a problem. Such a study is important, not only due to the fact that it allows us to estimate the robustness of the optimal solution that we have at hand, but also due to the insights it offers regarding the nature of the problem itself. There are several approaches to sensitivity analysis, the most common being the *tolerance* approach and the *parametric analysis* approach. In the tolerance approach one tries to find the interval in which the value of a given parameter must lie for the current optimal solution to remain optimal. In the parametric analysis approach, the optimal objective value is studied as a function of the value of some problem parameter that is varied from a lower bound to an upper bound. A third approach is the k-*best solutions* approach in which the best k solutions to a problem instance is output. This last approach has immense practical significance — it allows the decision maker to choose between "good" solutions on the basis of criteria that may be subjective in nature.

The earliest combinatorial optimization problems to be analyzed were the scheduling problem, the knapsack problem, the generalized assignment problem and the facility location problem (see Nauss [27]). The parametric analysis approach was used, and the problems were analyzed more as special cases of the general integer linear programming problem rather than as individual combinatorial problems. Geoffrion and Nauss [15] and Nauss [27] provide comprehensive surveys of sensitivity analysis results till the mid 1970's.

Individual combinatorial optimization problems have been actively studied since then. In the 1970's and '80's, the predominant approach was parametric analysis — a restricted list of publications would include Karp and Orlin [22] for shortest path problems, Jenkins [21] for fleet mix problems, Jenkins [20] for knapsack problems, Richter and Vörös [30] for lot-sizing problems, and Gusfield [18] for network flow problems. Some of the papers from this period using the tolerance approach were Gusfield [19], Shier and Witzgall [31], and Tarjan [38] on shortest path and network flow problems, and Sotskov [32] on scheduling.

The literature in the 1990's shows interesting trends. Most of the work reported seem to use the tolerance approach, and are concentrated on two problems — the traveling salesperson problem (see Libura [24], Sotskov *et al.* [34]) and the machine scheduling problem (see Bräsel *et al.* [4], Kravchenko *et al.* [23], Sotskov [33], Sotskov *et al.* [35], and Sotskov *et al.* [36]). Parametric analysis is used in Bunkard and Pfereschy [5], and

Fernández-Baca and Srinivasan [13]. The k-best approach is used in Van der Poort *et al.* [40] and Van der Poort *et al.* [41]. The related problem of stability analysis has also received much attention (see, among others, Chakravarti and Wagelmans [7], Bräsel *et al.* [4], Fernández-Baca and Srinivasan [13], Libura [24], Sotskov [33], Sotskov *et al.* [35], Sotskov *et al.* [36], Van der Poort [39] and Van der Poort *et al.* [42]).

The computational complexity of the sensitivity analysis problem for individual combinatorial optimization problems have also been studied. Cartensen [6] and Gusfield [19] have reported complexity results for parametric analysis applied to network flow problems. Cartensen [6], in particular showed that the analysis could (at least theoretically) be hard for some problems. Fernández-Baca and Slutzki [12] looked into the number of breakpoints in special cases of independent set problems and dominating set problems. Van Hoesel and Wagelmans calculated the complexity of determining tolerance of problem and performance parameters for the economic lot-sizing problems. Van der Poort [39] and Van der Poort *et al.* [42] reported extensive studies on the complexity of sensitivity analysis for the traveling salesperson problem.

For a comprehensive summary of work on sensitivity analysis published after 1977, we suggest that the interested reader refer to Greenberg [17] or van Hoesel *et al.* [43].

The problem of sensitivity analysis of generic combinatorial optimization problems and the complexity of such analysis have been address independently by Ramaswamy and Chakravarti [29] and Van Hoesel and Wagelmans [45]. In both papers, the cost coefficients of one of the elements in the ground set of the instance at hand is allowed to vary. Although the results obtained are very similar, there are important differences between the two papers. Ramaswamy and Chakravarti report results relating to the sensitivity analysis of optimal solutions to combinatorial optimization problems with both min-sum and min-max objectives, while Van Hoesel and Wagelmans consider tolerance limits for both optimal and ϵ-optimal solutions to combinatorial optimization problems but with min-sum objectives only. Again in Van Hoesel and Wagelmans [45] the cost coefficients are assumed to remain non-negative but Ramaswamy and Chakravarti [29] do away with this assumption.

It is rather surprising to note that almost all the literature is concerned with linear (or min-sum) objectives. In these problems, the cost of *all* the elements in a solution contribute towards the objective function, and so changes in the value of any element is immediately detected. Sensitivity analysis results therefore, even in the case of general combinatorial optimization problems with min-sum objectives, are easy applications of the results in Libura [24]. In combinatorial optimization problems with min-max objec-

tives, only a largest element contributes to the objective function. Sensitivity analysis of these problems are therefore more complicated.

Most of the published work on min-max problems is in Russian. Sotskov *et al.* [34] used the trajectory problem to provide a summary of this work. Gordeev and Leontev [16] report results on the stability aspects of combinatorial problems with min-max objectives. The current paper is an attempt to establish the complexity status of combinatorial problems with min-max objectives. In that direction it supersedes a part of Ramaswamy and Chakravarti [29].

We define a generic combinatorial optimization problem with min-max objective as follows.

Definition 1 *A combinatorial optimization problem (COP) with min-max objective is a collection of problem instances of the following form: we are given a finite ground set* G *of elements where each element* e_j *has a cost* c_{e_j}*, a collection* $F = \{f\}$ *of subsets of* G*, called feasible solutions (or simply solutions), and an objective function (or just objective)* $c(f) = \max_{e_j \in f}\{c_{e_j}\} : F \rightarrow \mathbb{R}$*. We are required to find an optimal feasible solution (or just optimum), i.e.,* $f_o \in F$ *such that* $c(f_o) = \min\{c(f) | f \in F\}$.

An instance of a COP is referred to as being *feasible* if $F \neq \emptyset$, and infeasible otherwise.

We make the following set of mild assumptions regarding COP's that will hold for the remainder of the paper.

1. Given an instance $I = (G, F, c)$ of a COP P and an arbitrary subset f of G, it is easy (i.e., possible in polynomial time) to check if $f \in F$.
2. It is easy (i.e., possible in polynomial time) to evaluate $c(f)$ for any solution f.
3. The empty set $\emptyset \in F$ and has an arbitrary cost which we denote by $c(\emptyset)$. It is also easy to evaluate $c(\emptyset)$ whenever $\emptyset \in F$. The time required to evaluate $c(\emptyset)$ will be denoted by $T_\emptyset$.
4. It is possible to find some feasible solution to each instance I of P in polynomial time.
5. Given an instance $I = (G, F, c)$ of a polynomially solvable COP P, all instances $I' = (G', F', c)$ in which $G' \subseteq G$ and $F' \subseteq F$ are polynomially solvable.

Assumptions 1 and 4 are valid for most common COP's. The time required to compute the objective of a given solution f is at worst $|f|$ for min-max objectives, and so this assumption is in fact trivial. Assumption 3 involves no loss in generality — if $\emptyset \notin F$

then we can set $c(\emptyset) = \infty$. Assumption 5 is an implicit assumption in all work on this area. We list it here for the sake of completeness.

We will use the tolerance approach to sensitivity analysis in this paper. Recall that in the tolerance approach we find the interval within which a problem parameter may vary so that the current optimal solution remains optimal. This interval is commonly described in the form of *exact upper and lower tolerances*. We will formally define these tolerances and the sensitivity analysis problem as follows.

Definition 2 *Given an element e in an instance $I = (G, F, c)$ of a COP P and an optimal solution f_o to I, the **exact upper tolerance (EUT)** β_e is defined as*

$$\beta_e = \sup\{\delta | f_o \textit{ remains optimal when } c_e \rightarrow c_e + \delta\}.$$

*The **exact lower tolerance (ELT)** α_e is defined as*

$$\alpha_e = \sup\{\delta | f_o \textit{ remains optimal when } c_e \rightarrow c_e - \delta\}.$$

Definition 3 *The sensitivity analysis of a COP P is defined as follows.*

Problem	*SA(P): Sensitivity Analysis for COP P*
Input	*Instance $I = (G, F, c)$ of P, optimal solution f_o to I*
Output	*α_e and β_e for each $e \in G$.*

Notice that SA(P) consists of the following two component problems, corresponding to each $e \in G$.

Problem	LTOL(P): Lower Tolerance for COP P
Input	Instance $I = (G, F, c)$ of P, optimal solution f_o to I, $e \in G$
Output	α_e.

Problem	UTOL(P): Upper Tolerance for COP P
Input	Instance $I = (G, F, c)$ of P, optimal solution f_o to I, $e \in G$
Output	β_e.

Obviously SA(P) is polynomially solvable, if and only if LTOL(P) and UTOL(P) are both polynomially solvable for each $e \in G$.

In this paper, in Section 2 we provide characterizations of the ELT and EUT for an arbitrary element of the ground set of min-max COPs. We then proceed in Section 3 to show that under very mild assumptions, the sensitivity analysis problem SA(P)

is about as hard as the COP P itself, in the sense that polynomial solvability of P implies polynomial solvability of SA(P) and vice-versa. We use the example below in Sections 2 and 3 to illustrate our arguments and algorithms. Finally Section 4 is a brief concluding section.

Example 1. We consider an instance of a symmetric non-Euclidean bottleneck traveling salesperson problem (BTSP) with the following distance matrix (c).

$\mathbf{c}(.)$	0	1	2	3	4	5
0	–	14	2	6	17	15
1	14	–	4	18	16	6
2	2	4	–	11	6	8
3	6	18	11	–	15	1
4	17	16	6	15	–	17
5	15	6	8	1	17	–

In this example, the edges $(u,v), u \neq v$ are the elements of the problem, $G = \{(0,1), (0,2), \ldots, (0,5), (1,2), \ldots, (4,5)\}$. Any solution f is a collection of edges that form a TSP tour, so that F is a collection of all TSP tours. The objective function c for a solution $f \in F$ is the cost of the longest edge in f. The optimal solution that we consider here is $f_o = \{(0,3), (3,4), (2,4), (1,2), (1,5), (0,5)\}$, so that $c(f_o) = c(3,4) = 15$. ■

2. Characterizations of α_e and β_e for the min-max objective

In this section we derive characterizations of the ELT and EUT for elements in COP's with min-max objectives.

Let $f = \{e_1, e_2, \ldots, e_r\} \in F$ be a feasible solution, where $c_{e_1} \geq c_{e_2} \geq \cdots \geq c_{e_r}$. We call each element $e \in f$ with $c_e = c_{e_1}$ a *largest* element of f. Any element $e' \in f$ with $c_{e'} = c_{e_2}$ is called a *second largest* element of f and denoted by $c^{(2)}(f)$. If $|f| = 1$, then $c^{(2)}(f) = -\infty$. Note that there may be more than one largest and second largest element in a given solution, and that a largest element may have the same cost as a second largest element. Given $e \in G$ and $f \in F$, we say that f is *e-critical* if $e \in f$ and $c(f) = c_e$, and that f is a *best* e-critical solution if it is e-critical and no other e-critical solution f' has $c^{(2)}(f') < c^{(2)}(f)$. $P_e = \{p_e\}$ denotes the set of best e-critical solutions, and $P^e = \{p^e\}$ denotes set of best solutions not containing e.

Example 2. (running example) In addition to f_o, there are three other solutions with an objective of 15. These are $f_1 = \{(0,1),(1,5),(2,5),(2,4),(3,4),(0,3)\}$, $f_2 = \{(0,1),(1,5),(3,5),(3,4),(2,4),(0,2)\}$, and $f_3 = \{(0,1),(1,2),(2,4),(3,4),(3,5),(0,5)\}$. Notice that $c(f_o) = c(f_1) = c(f_2) = c(f_3) = 15$, but $c^{(2)}(f_o) = c^{(2)}(f_3) = 15$, while $c^{(2)}(f_1) = c^{(2)}(f_2) = 14$. If we consider $e = (0,5)$, then f_o and f_3 are e-critical but f_1 and f_2 are not. Further, both f_0 and f_3 are best e-critical solutions. So $P_e = \{f_0, f_3\}$ and $P^e = \{f_1, f_2\}$. However, if $e = (3,4)$, then f_o through f_3 are all e-critical but f_1 and f_2 are the only two best e-critical solutions. ■

In order to derive expressions for the exact tolerances of elements $e \in G$, we need to study the objective function $c(f)$ as a function of c_e. It is trivial to see that $c(f)$ is unaffected by changes in c_e if $e \notin f$. If $e \in f$, then there are two cases to consider. In case f is e-critical, i.e. $c(f) = c_e$, then an increase in c_e would cause an equal increase in $c(f)$. A decrease in c_e by an amount not more than $c(f) - c^{(2)}(f)$ would decrease $c(f)$ by an equal amount. If c_e decreases further, f ceases to be e-critical, and $c(f)$ remains constant. In case f is not e-critical, a decrease in c_e, or an increase in c_e by an amount not more than $c(f) - c_e$ will not affect $c(f)$. If c_e increases by an amount more than $c(f) - c_e$, then f becomes e-critical and increases linearly with c_e. Figure 1 shows $c(f)$ as a function of c_e.

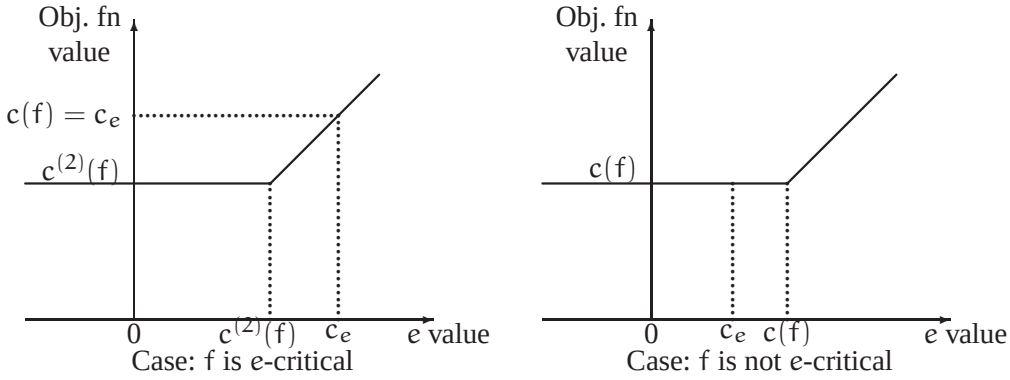

Figure 1: $c(f)$ as a function of the c_e when $e \in f$

We now characterize the exact tolerances of an arbitrary element e i.e., its α_e and β_e values. In the proofs of the two propositions, we will consider cost transformations in

which c_e is increased or decreased by an amount δ. In each case the cost of a solution f as a result of the transformation will be denoted by $c_\delta(f)$.

Proposition 1 *(Exact lower tolerances)*

$$\alpha_e = \begin{cases} c_e - c^{(2)}(f_o) & \text{if } e \in f_o,\ c(f_o) = c_e,\ \exists g \in F \text{ such that } e \in g, c(g) = c_e, \\ & \text{and } c^{(2)}(g) < c^{(2)}(f_o); \\ c_e - c(f_o) & \text{if } e \notin f_o,\ c(f_o) \leq c_e,\ \exists h \in F \text{ such that } e \in h, c(h) = c_e, \\ & \text{and } c^{(2)}(h) < c(f_o); \\ \infty & \text{otherwise.} \end{cases}$$

PROOF. Consider an element $e \in G$. If $c_e < c(f_o)$, then clearly there are no e-critical solutions. Reducing c_e for such elements cannot affect the optimality of f_o. So α_e is trivially ∞ for all elements e with $c_e < c(f_o)$. In the remainder of this proof therefore, we consider only those elements with $c_e \geq c(f_o)$.

Consider the cost transformation $c_e \rightarrow c_e - \delta, \delta > 0$. From Figure 2 it is clear that if $e \in f_o$ and $c(f_o) = c_e$ then

$$c_\delta(f_o) = \begin{cases} c(f_o) - \delta & \text{if } \delta \leq c(f_o) - c^{(2)}(f_o); \\ c^{(2)}(f_o) & \text{otherwise.} \end{cases}$$

If $\exists g \in F$ such that $e \in g$, $c(g) = c_e$ and $c^{(2)}(g) < c^{(2)}(f_o)$, then if $\delta > c_e - c^{(2)}(f_o)$ then $c_\delta(f_o) = c^{(2)}(g) - \delta$, which affects the optimality of f_o. If no such g exists, or if $c(f_o) > c_e$ (in which case $c_\delta(f_o) = c(f_o)$), then δ can be arbitrarily large.

Next consider the same transformation but assume that $e \notin f_o$, and $c(f_o) \leq c_e$. It is clear that $c_\delta(f_o) = c(f_o)$. If $\exists h \in F$ such that $c(h) = c_e$ then

$$c_\delta(h) = \begin{cases} c(h) - \delta & \text{if } \delta \leq c(h) - c^{(2)}(h); \\ c^{(2)}(h) & \text{otherwise.} \end{cases}$$

If $c^{(2)}(h) < c(f_o)$, then if $\delta > c_e - c(f_o)$, then $c_\delta(h) < c_\delta(f_o)$ which compromises the optimality of f_o. If no such h exists, or if $c(f_o) > c_e$, δ can be arbitrarily large.

The proposition follows. □

Proposition 2 *(Exact upper tolerances)*

$$\beta_e = \begin{cases} \{c(p^e) - c_e\} & \text{if } e \in f_o; \\ \infty & \text{otherwise.} \end{cases}$$

PROOF. Consider an arbitrary element $e \in f_o$. There are two cases to consider.

Case 1: $c(f_o) = c_e$. Let $c_e \rightarrow c_e + \delta, \delta > 0$; then $c_\delta(f_o) = c(f_o) + \delta$. Note that since e is a largest element of f_o, no solution containing e can be superior to f_o. Thus f_o is no longer optimal if and only if $\exists p^e \in P^e$ such that $\delta > c(p^e) - c_e$. In this case the minimum value of δ would be $\{c(p^e) - c_e\}$.

Case 2: $c(f_o) > c_e$. In this case we may increase c_e by an amount $\delta_1 = c(f_o) - c_e$ without changing the cost of any solution. The rest of the analysis is exactly as in Case 1 above, i.e. we may further increase c_e by exactly $\delta_2 = \{c(p^e) - c(f_o)\}$ without violating the optimality of f_o. The total permissible increase before f_o becomes non-optimal is thus $\delta_1 + \delta_2 = \{c(p^e) - c_e\}$ in this case.

If $e \notin f_o$, increasing c_e leaves $c(f_o)$ unchanged. Since such an increase cannot cause the cost of any solution to decrease, c_e can be increased indefinitely without affecting the optimality of f_o.

The proposition follows. □

Example 3. (running example) Let us suppose that we want to calculate the ELT and EUT for edges $(3,4)$ and $(4,5)$.

In case of edge $(3,4) \in f_o$, $c_{(3,4)} = 15 = c(f_o)$. Note that both f_1 and f_2 contain this edge and $c^{(2)}(f_1), c^{(2)}(f_2) < c^{(2)}(f_o)$. Therefore either f_1 or f_2 can be the solution g mentioned in Proposition 1, and $\alpha_{(3,4)} = c_{(3,4)} - c^{(2)}(f_1) = 15 - 14 = 1$. If $c_{(3,4)}$ decreases by more than 1, then f_1 (and also f_2) become optimal and f_o becomes suboptimal. In our example, the solution $f = \{(0,1),(1,4),(4,2),(2,3),(3,5),(5,0)\}$ is a member of P^e and so $\beta_{(3,4)} = c(f) - c_{(3,4)} = 16 - 15 = 1$. If $c_{(3,4)}$ increases by more than 1, then f becomes the new optimal solution.

In case of edge $(4,5) \notin f_o$, $c_{(4,5)} = 17 > c(f_o)$. In our example we indeed have a solution $f = \{(0,3),(3,2),(2,4),(4,5),(5,1),(1,0)\}$ with $c^{(2)}(f) = 14 < c(f_o)$. So $\alpha_{(4,5)} = c_{(4,5)} - c(f_o) = 17 - 15 = 2$. If $c_{(4,5)}$ reduces by more than this amount, f becomes the new optimal solution. Since $(4,5) \notin f_o$, $c_{(4,5)}$ can increase indefinitely without affecting the optimality of f_o, which means $\beta_{(4,5)} = \infty$. ■

From Proposition 1 we see that LTOL is non-trivial in COP's with min-max objectives, only for elements $e \in G$ with $c_e \geq c(f_o)$. Given such an element e, if $e \in f_o$, it is necessary and sufficient to determine if $\exists f \in F$ with $e \in f$, $c(f) = c_e$ and $c^{(2)}(f) < c^{(2)} f_o$ in order to determine if α_e is finite. If it is finite, then it is necessary and sufficient

to determine $c^{(2)}(f_o)$ to determine the value of α_e. Hence solving LTOL for $e \in f_o$ is equivalent to solving the following problem.

Problem	LM1 : First Lower Tolerance Problem for P, Min-max Objective
Input	Instance $I = (G, F, c)$ of COP P; optimum solution f_o to I; $e \in f_o$
Output	"YES" if $\exists f \in F$ with $e \in f$, $c(f) = c_e$ and $c^{(2)}(f) < c^{(2)}(f_o)$; "NO" otherwise.

Given, on the other hand, an element $e \in G \setminus f_o$ with $c_e \geq c(f_o)$, we see that to determine α_e, it is necessary and sufficient to determine if $\exists f \in F$ with $e \in f$, $c(f) = c_e$ and $c^{(2)}(f) < c(f_o)$. Solving LTOL for $e \in G \setminus f_o$ is thus equivalent to solving the following problem.

Problem	LM2 : Second Lower Tolerance Problem for P, Min-max Objective
Input	Instance $I = (G, F, c)$ of COP P; optimum solution f_o to I; $e \in G \setminus f_o$
Output	"YES" if $\exists f \in F$ with $e \in f$, $c(f) = c_e$ and $c^{(2)}(f) < c(f_o)$; "NO" otherwise.

We next formulate below an evaluation problem LM, such that the polynomial solvability of LM implies the polynomial solvability of both LM1 and LM2.

Problem	LM : Evaluation Version of LM1 and LM2
Input	Instance $I = (G, F, c)$ of COP P; optimum solution f_o to I; $e \in G$
Output	$c^{(2)}(p_e)$ if $\alpha_e < \infty$; ∞ otherwise.

Note that we do not include $c^{(2)}(f_o)$ in the output of LM. In the optimization version of LM, the output would be a best e-critical solution if $\alpha_e < \infty$, and ∞ otherwise.

We can conclude from Proposition 2 that UTOL is non-trivial only for elements of f_o. For each such element e it is necessary and sufficient to determine $c(p^e)$ to be able to determine β_e. Solving UTOL for $e \in f_o$ is thus equivalent to solving the following problem.

Problem	UM : Equivalent Upper Tolerance Problem for P, Min-max Objective
Input	Instance $I = (G, F, c)$ of COP P; optimum solution f_o to I; $e \in f_o$
Output	$c(p^e)$.

3. Complexity of SA for min-max problems

In this section we explore the relationship between polynomial solvability of P and the polynomial solvability of SA(P).

First we assume that we have a polynomial time algorithm SOLVE_P to solve P. (SOLVE_P is assumed to return ∞ if P has no solution.) We will now show that we can use this algorithm to obtain polynomial-time solution algorithms LM_VIA_P and UM_VIA_P for LM and UM respectively. Given $e \in G$, algorithm LM_VIA_P listed below polynomially solves LM by examining the effect of lowering c_e to a very low value on the optimality of f_o.

```
Algorithm LM_VIA_P
Input:    Instance I = (G, F, c) of COP P; an optimal solution f_o of I; e ∈ G
Output:   c^(2)(p_e) if α_e < ∞; ∞ otherwise
begin
          c_o ← c(f_o);
          M ← min_{e∈G}{c_e}; /* M is the cost of a smallest element of G */
          s ← c_e; /* store the cost of e */
a1:       c_e ← M − 1;
          call SOLVE_P to find C, the cost of the best solution of I;
          if C < c(f_o) then
                    if C = M − 1 then
                    begin
                              c_e ← s; /* restoring the cost of c_e */
                              return −∞;
                    end
                    else
                    begin
                              c_e ← s; /* restoring the cost of c_e */
                              return C;
                    end
          else
                    begin
                              c_e ← s; /* restoring the cost of c_e */
                              return ∞;
                    end
end.
```

Example 4. (running example) We will illustrate the working of some of the cases in this algorithm, other cases can be worked through in a similar manner.

Let us input the distance matrix, f_o, and the edge $(0,3)$ to LM_VIA_P. $M = c_{\{(3,5)\}} = 1$. But since there are no $(0,3)$-critical solutions, $C = 15$ even when $c_{(0,3)}$ is set to 0. Hence LM_VIA_P returns ∞.

Let us suppose that we input the edge $(3,4)$ instead of $(0,3)$. In this case, too $M = c_{(3,5)} = 1$. If $c_{(3,4)}$ is set to 0, $c(f_o)$ still remains 15 (due to the presence of the edge $(0,5) \in f_o$. Thus $C = 15 = c(f_o)$ and so LM_VIA_P returns ∞. An inspection of f_o shows that $\alpha_{(3,4)} = \infty$, precisely due to the edge $(0,5) \in f_o$).

Let us finally input the edge $(4,5)$ (not in f_o) instead. Here too $M = c_{(3,5)} = 1$. If $c_{(4,5)}$ is set to 0, SOLVE_P causes C to be set to 14 (refer to the previous portion of this example on page 6). Therefore LM_VIA_P returns 14 in this case, which actually is $c^{(2)}(p_{(4,5)}$. ■

The following theorem shows that this algorithm is correct and polynomial.

Theorem 1 *Algorithm LM_VIA_P correctly solves LM in polynomial time.*

PROOF. From Proposition 1 it follows that α_e, if finite, cannot exceed $c_e - M$, and hence that the cost transformation in Step a1 affects the optimality of f_o if and only if $\alpha_e < \infty$. The output produced is clearly correct if $\alpha_e = \infty$. If $\alpha_e < \infty$ then $\alpha_e \leq c_e - c^{(2)}(f_o) < c_e - M + 1$, so that the new optimum must be a solution p_e in P_e, with a cost equal to that of the largest element in $P_e \setminus \{e\}$, which is just the value of $c^{(2)}(p_e)$ at the outset of the algorithm. Finally, C equals $M - 1$ if and only if the solution whose cost is found by SOLVE_P is $\{e\}$. In this case $c^{(2)}(p_e) = -\infty$ by definition. Correctness of the algorithm follows.

Noting that computing M requires $O(|G|)$ time, and assuming that SOLVE_P requires polynomial time, we conclude that the running time of LM_VIA_P is polynomial.

□

In a similar fashion, we may easily devise algorithm UM_VIA_P which, given any element $e \in G$, uses SOLVE_P to polynomially find the cost of the best solution not containing e by setting c_e to a very high value so that the best solutions to P are precisely those in P^e.

Thus we see that if P is polynomially solvable, then LM and UM are also polynomially solvable. This gives us the following result.

Theorem 2 *Let P be a COP with a min-max objective. If P is polynomially solvable, then SA is also polynomially solvable.*

We next assume that SA is polynomially solvable, i.e. we have available polynomial time solution algorithms for LM1 and LM2. We will use these to obtain a polynomial time solution algorithm for P, under the weak assumption 4 in the introductory section. Recall that this assumption, which will remain in force for the remainder of this section, states that each instance of P is feasible and that it is possible to find some feasible solution in polynomial time. It is unnecessary to assume the existence of a polynomial time solution algorithm for UM. Our approach is to begin by constructing a polynomial time algorithm for solving the optimization version of LM, which we then use to construct a polynomial time algorithm for P.

We transform the costs in the following manner: assign each smallest element a cost 1, each next larger element a cost 2 and so on. We will refer to this transformation as Ψ and refer to the cost of an edge e after this transformation as $c_{\Psi e}$. Ψ can be completed in $O(|G|\log|G|)$ time and clearly preserves optimal solutions.

Lemma 1 *Let $I = (G, F, c)$ be a min-max COP with optimum f_o. For an arbitrary but fixed number $L < c(f_o)$, consider the cost transformation $T : c_{e'} \rightarrow L$ for each $e' \in f_o$ with $c_{e'} > L$. Then*

1. *T leaves f_o optimal with cost L;*
2. *for each $e \in G$ with $\alpha_e < \infty$ before T, $\alpha_e < \infty$ after T if and only if $L > c^{(2)}(p_e)$.*

PROOF. 1. Trivial.

2. Consider an element $e \in G$ for which $\alpha_e < \infty$. Denote by $c_T^{(2)}(f)$ the cost, after T, of a second largest element of a solution f. There are two cases to consider.

   Case 1: $e \in f_o$. Since $\alpha_e < \infty$ before T, it follows from Proposition 1 that $c_e = c(f_o)$ and that $c^{(2)}(p_e) < c^{(2)}(f_o)$. If $L > c^{(2)}(p_e)$, then, since $L < c(f_o) = c(p_e)$ we have $c_T(p_e) = L$ and $c^{(2)}(p_e) = c_T^{(2)}(p_e) < c_T^{(2)}(f_o)$. From Proposition 1 we conclude that α_e is finite.

   To prove the converse, suppose $L \leq c^{(2)}(p_e)$. Then $c_T^{(2)}(f_o) = L$, since $\alpha_e < \infty$ and so $c^{(2)}(p_e) < c^{(2)}(f_o)$ which implies that $L < c^{(2)}(f_o)$. However, $c_T^{(2)}(p_e) \geq L$ and the result follows from Proposition 1.

   Case 2: $e \notin f_o$. Since $\alpha_e < \infty$ before T, it follows from Proposition 1 that $c^{(2)}(p_e) < c(f_o)$. We also have $c_T(p_e) = c_e$. If $L > c^{(2)}(p_e)$, then $c^{(2)}(p_e) = c_T^{(2)}(p_e) < c_T(f_o) = L$. From Proposition 1 we conclude that

α_e is finite. To prove the converse, suppose $L < c^{(2)}(p_e)$. Then $c_T(f_o) = L \leq c_T^{(2)}(p_e)$ and the result follows from Proposition 1.

□

Algorithm OPT_LM below solves the optimization version of LM in polynomial time, given polynomial algorithms SOLVE_LM1 and SOLVE_LM2 for LM1 and LM2 respectively. It uses a two phase procedure. In Phase 1, Lemma 1 is used to determine $c^{(2)}(p_e)$. In Phase 2, we examine the effect of suitably raising $c_{e'}$ for each $e' \in G$ with $c_{e'} \leq c^{(2)}(p_e)$ (these are the only candidate elements of p_e) on α_e: if α_e remains unchanged, then we may look for a solution in P_e not containing e'; if α_e increases (it actually becomes infinite), then e' must belong to each best solution containing e.

We present the algorithm for the case $e \in f_o$; for the case $e \notin f_o$, "SOLVE_LM1" is replaced by "SOLVE_LM2" throughout. In order to improve the readability of the pseudocode, we will assume that the cost transformation Ψ has already been applied to the element costs before invoking this algorithm.

Algorithm OPT_LM
Input: Instance $I = (G, F, c)$ of COP P; optimum solution f_o to I; $e \in G$
Output: p_e if $\alpha_e < \infty$; $\emptyset$ otherwise
Assumption: The element costs have already been transformed by the cost transformation Ψ
begin

```
/* begin preprocessing */
if SOLVE_LM1(P, f_o, e) returns "NO" then
        return ∅; /* α_e is infinite */
/* end preprocessing */
/* begin initialization process */
for e' ∈ G do
        s_e' ← c_e'; /* storing the cost vector */
IN ← {e};
OUT ← ∅;
LIST ← ∅;
L ← c(f_o);
/* end initialization process */
for each e' ∈ G do
        k_e' ← c_e'; /* store the value of c_e' */
```

```
Phase 1:  L ← L − 1;
f1:       for each e′ ∈ f_o with c_e′ > L do
          begin
                    c_e′ ← L;
                    LIST ← LIST + e′;
          end
          if SOLVE_LM1(P, f_o, e) returns "NO" then
                    go to Phase 2; /* c^(2)(p_e) = L */
          else
                    go to Phase 1;
Phase 2:  for each e′ in LIST do
                    c_e′ ← k_e′; /* restore original costs */
f2:       for each e′ ∈ G with c_e′ ≤ L /* these are the only candidate elements of p_e \ {e} */
          begin
                    c_e′ ← c(f_o); /* f_o remains optimal */
                    if SOLVE_LM1(P, f_o, e′) returns "NO" then
                    begin
                              IN ← IN + e′;
                              c_e′ ← k_e′;
                    end
                    else
                              OUT ← OUT + e′;
          end
          return IN;
end.
```

Example 5. (running example) We illustrate the working of OPT_LM. After the cost transformation Ψ, the distance matrix (c_Ψ) looks as below.

$\mathbf{c}_\Psi(.)$	0	1	2	3	4	5
0	–	7	2	4	10	8
1	7	–	3	11	9	4
2	2	3	–	6	4	5
3	4	11	6	–	8	1
4	10	9	4	8	–	10
5	8	4	5	1	10	–

Let us assume we input f_o and edge $(4,5)$ to OPT_LM. Note that $\alpha_{(4,5)} = 2$. Initially $IN = \{(4,5)\}, OUT = LIST = \emptyset$, and L is set to 10. The assumed polynomial algorithm SOLVE_LM2 confirms that $\alpha_{(4,5)}$ is indeed finite and we start off with Phase 1. At the end of Phase 1, $L = 7$, and $LIST = \{(0,4),(0,5),(1,3),(1,4),(3,4),(4,5)\}$.

In Phase 2, we first restore the costs of the members of LIST to their c_Ψ values. This step is strictly not necessary for the correctness of this algorithm, but is essential if this algorithm is being called by any other. Let us consider the Phase 2 process for the element $(0,1)$ ($c_{\Psi(0,1)} = 7 \leq L$). If $c_{\Psi(0,1)} \rightarrow 8$, then SOLVE_LM2 will return ∞, since none of the $(4,5)$-critical solutions have their second largest elements less than 8 after this transformation. Hence $(0,1)$ gets added to IN. If we repeat the Phase 2 procedure for all the elements not in LIST, we end up with IN $= \{(0,1),(0,3),(1,5),(2,3),(2,4),(4,5)\}$ which is a best $(4,5)$-critical solution. ■

The next theorem shows that OPT_LM is correct and polynomial.

Theorem 3 *Algorithm OPT_LM correctly solves the optimization version of LM in polynomial time.*

PROOF. We first verify the correctness of the algorithm. The case $\alpha_e = \infty$ is trivial. Consider the case $\alpha_e < \infty$. Correctness of Phase 1 (i.e., that L equals $c^{(2)}(p_e)$ at the end of Phase 1) follows from Lemma 1; it suffices therefore to prove the correctness of Phase 2. We prove by induction that at each execution of the *for* loop f2, there is a solution in P_e that contains each element of IN and no element of OUT. This is trivially true at the beginning of f2 when IN $= \{e\}$ and OUT $= \emptyset$, since $P_e \neq \emptyset$. Assume the induction hypothesis to be true at some stage of f2, and let e' be the next element to undergo the transformation $c_{e'} \leftarrow c(f_o)$. Since $c_{e'} < c(f_o)$, this transformation leaves f_o optimal. Thus SOLVE_LM1 (SOLVE_LM2 if $e \notin f_o$) correctly predicts whether α_e is finite. Since α_e was finite before the transformation and since each element of OUT has cost $c(f_o)$, it follows that α_e is now infinite if and only if each solution in P_e contains an element of OUT $+ e'$. Hence the induction hypothesis holds at the end of f2. At termination, however, OUT is identical with $G \setminus$ IN. Correctness of OPT_LM follows.

We turn now to the running time. Since $c(f_o) \leq |G|$, by virtue of the cost transformation Ψ, the *go to* statement is executed no more than $|G|$ times. Each *for* loop in the algorithm loops $O(|G|)$ times at each invocation. Phase 1 thus takes $O(|G|^2)$ time. In Phase 2, either SOLVE_LM1 or SOLVE_LM2 is called once during each execution of f2. Phase 2 thus takes $O(|G|R)$ time where $O(R)$ is the greater of the assumed running times of SOLVE_LM1 and SOLVE_LM2. So OPT_LM runs in time $O(|G|\{R + \log|G|\})$ and the proof is complete. □

We now show that COP P can be solved polynomially using the polynomial algorithms SOLVE_LM1, SOLVE_LM2 and OPT_LM. The idea behind the proposed algorithm is the following. We start with a feasible solution f_o of the instance I of P, and make it optimal by raising the cost of each $e \in G$ with $c_e < c(f_o)$, to $c(f_o)$. Next, for each element e whose cost was thus altered, we determined if lowering its cost to its original value will violate the optimality of f_o. If it will, then we find p_e using OPT_LM, lower c_e to its original value, and declare p_e as the new optimal solution. Proceeding this way, we restore all element costs to their original values, and terminate with an optimal solution to I.

```
Algorithm P_via_LM
Input:    Instance I = (G, F, c) of COP P
Output:   A best solution of I
begin
          LIST ← ∅;
          find a feasible solution f_o of I; /* assumed possible in polynomial time */
l1        for each e ∈ G with c_e < c(f_o) do
          begin
                    LIST ← LIST + e;
                    k_e ← c_e;
                    c_e ← c(f_o); /* f_o is now an optimal solution of I */
          end
l2        for each e in LIST do
          begin
                    call SOLVE_LM and find α_e;
                    if α_e ≥ c(f_o) − k_e then
                              c_e ← k_e; /* f_o remains optimal */
                    else
                    begin
                              call OPT_LM to find p_e, a best e-critical solution;
                              c_e ← k_e;
                              f_o ← p_e; /* p_e is now an optimal solution */
                    end
          end
          return f_o;
end.
```

Example 6. An initial feasible solution is easy to find for the BTSP in our example. We will choose the solution $f_o = \{(0,1),(1,2),(2,3),(3,4),(4,5),(0,5)\}$ with cost $c(f_o) = 17$. At the beginning of the *for* loop l2 therefore, LIST $= \{(0,1),(0,2),(0,3),(0,5),(1,2),\ (1,4),(1,5),(2,3),(2,4),(2,5),(3,4),(3,5)\}$. Assuming that l2 picks

out elements in the same order, the costs of $(0,1)$ through $(2,5)$ are restored to their initial values, since α_e for any of these edges is infinite. When P_via_LM considers the edge $(3,4)$, it finds that $\alpha_{(3,4)} < 17 - 15 = 2$ and calls OPT_LM after setting $c_{(3,4)}$ to 15. OPT_LM returns with the solution $\{(0,1),(1,5),(5,2),(2,4),(4,3),(3,0)\}$ which becomes the new f_o (with cost 15). This is an optimal solution for the example at hand, and remains unchanged when $c_{(3,5)}$ is considered and restored to its original value.

■

Theorem 4 *Let* P *be a COP with a min-max objective. If LM1 and LM2 are polynomially solvable, then* P *is polynomially solvable whenever a feasible solution to each instance* I *of* P *can be found in polynomial time.*

PROOF. We first prove that the solution f_o returned by P_via_LM is indeed an optimal solution to I. The proof is by induction. The algorithm generates a sequence $\{I_1, I_2, \ldots, I\}$ of instances of P, terminating in the original instance I. Assume that at some stage of the algorithm, f_o is the optimal solution to the current instance I_k. This is trivially true for the case $k = 1$ since the cost of each element in G is $\geq c(f_o)$. Let e be the element picked from LIST in the *for* loop l2. If $\alpha_e \geq c(f_o) - k_e$ then f_o remains optimal on decreasing c_e from $c(f_o)$ to k_e. Suppose on the other hand $\alpha_e < c(f_o) - k_e$. This implies, that decreasing c_e from its present value $c(f_o)$ to k_e will render p_e optimal, i.e., an optimum of I_{k+1}. The result follows by induction.

Since the total number of calls made to each of SOLVE_LM and OPT_LM by Algorithm P_via_LM is clearly no more than $|G|$, P_via_LM is a polynomial algorithm if LM1 and LM2 are polynomially solvable. □

This leads us to the following important corollary.

Corollary 3.1 *If SA is polynomially solvable, then* P *is polynomially solvable whenever a feasible solution to each instance* I *of* P *can be found in polynomial time.*

4. Conclusion

In this paper we have shown that under the weak assumptions that testing feasibility and evaluating a feasible solution is easy, the sensitivity analysis problem (viz., the problem of determining exact upper and lower tolerances) for an arbitrary combinatorial

optimization problem P with a min-max objective is easy, i.e., polynomially solvable, if the original combinatorial optimization problem P is itself easy. Our proofs are constructive and provide a polynomial method for solving the sensitivity analysis problem whenever a polynomial time algorithm for solving P is available. Better methods may of course be available for determining tolerances for specific combinatorial optimization problems. We have also shown that under the additional assumption that it is easy to determine an initial feasible solution, sensitivity analysis is easy only if P is easy. We have illustrated all our results using an instance of a non-Euclidean bottleneck traveling salesperson problem as an example.

Our results imply that unless $\mathcal{P} = \mathcal{NP}$ polynomial time algorithms for sensitivity analysis for many well-known $\mathcal{NP}$-hard combinatorial optimization problems such as the bottleneck traveling salesperson problem is not possible.

Acknowledgement

The authors thank Dr. B. Goldengorin for his suggestions which have helped to improve the presentation of the results in this paper.

References

[1] M. O. Ball and R. Taverna, Sensitivity analysis in matching and its use in solving matching problems with a single side constraint, Annals of Operations Research, 4, (1985). 25-56

[2] C.E. Blair. Sensitivity analysis for knapsack problems: A negative result, Discrete Applied Mathematics, 81(1–3), (1998). 133–139

[3] H. Booth and J. Westbrook. Alinear algorithm for analysis of minimum spanning and shortest path trees of planar graphs, Algorithmica, 11, (1994). 341–352

[4] H. Bräsel, Y.N. Sotskov and F. Werner. Stability of a schedule minimizing mean flow time, Mathematical Computer Modelling, 24(10), (1996). 39–53

[5] R.E. Bunkard and U. Pferschey. The inverse-parametric knapsack problem, European Journal of Operational Research, 83(2), (1995). 376–393

[6] P.J. Cartensen. Complexity of some parametric integer and network programming problems, Mathematical Programming, 26(1), (1983). 64–75

[7] N. Chakravarti and A.P.M. Wagelmans, Calculation of stability radii for combinatorial optimization problems, Operations Research Letters, 23, 1998. 1–7

[8] N. Chakravarti, Sensitivity Analysis in Isotonic Regression, Discrete Applied Mathematics, 45, (1993). 183–196
[9] F.Y. Chin and D.J. Houck, Algorithms for updating minimal spanning trees, Journal of Computer & System Sciences, 16, (1978). 333–344
[10] W. Cook, A.M.H. Gerrards, A. Schrijver and E. Tardos, Sensitivity theorems in integer linear programming, Mathematical Programming, 34, (1986). 251–264
[11] U. Derigs. Postoptimality analysis for matching problems, Mathematics of Operations Research, 49, (1985). 215–221
[12] D. Fernández-Baca and G. Slutzki. Solving parametric problems on trees, Journal of algorithms, 10, (1989). 381–402
[13] D. Fernández-Baca and S. Srinivasan. Constructing the minimization diagram of a two-parameter problem, Operations Research Letters, 10, (1991). 87–93
[14] T. Gal and H.J. Greenberg. Advances in sensitivity analysis and parametric programming, Kluwer Academic Publishers, Boston, (1997)
[15] A. M. Geoffrion and R. Nauss. Parametric and postoptimality analysis in integer linear programming, Management Science, 23(5), (1977). 453-466
[16] E.N. Gordeev and V.K.Leontev. Stability in bottleneck problems, Computational Mathematics and Mathematical Physics, 20(4), (1981). 275–280
[17] H. J. Greenberg. An annotated bibliography for post-solution analysis in mixed integer programming and combinatorial optimization, in D.L. Woodruff (ed.), Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search, Kluwer Academic Publishers, (1998). 97–148
[18] D. Gusfield. Sensitivity analysis for combinatorial optimization, Ph.D. thesis, Electronics Research Laboratory, UC Berkeley, CA, (1980).
[19] D. Gusfield. A note on arc-tolerances in sparse shortest path and network flow problems, Networks, 13(2), (1983). 191–196
[20] L. Jenkins. Parametric-objective integer programming using knapsack facets and Gomory cutting lanes, European Journal of Operational Research, 31(1), (1987). 102–109
[21] L. Jenkins. Using parametric integer programming to plan the mix of an air transport fleet, INFOR, 25(2), (1987). 117–135
[22] R.M. Karp and J.B. Orlin. Parametric shortest path algorithms with an application to cyclic staffing, Discrete Applied Mathematics, 3, (1981). 37–45
[23] S.A. Kravchenko, Y.N. Sotskov and F. Werner. Optimal schedules with infinitely large stability radius, Optimization, 33, (1995). 271–280
[24] M. Libura. Sensitivity analysis for minimum Hamiltonian path and traveling salesman problems, Discrete Applied Mathematics, 30(2), (1991). 197–211

[25] M. Libura. Optimality conditions and sensitivity analysis for combinatorial optimization problems, Control and cybernetics, 25(6), (1996). 1165–1180
[26] M. Libura. On accuracy of solutions for discrete optimization problems with perturbed coefficients of the objective function, Annals of Operations Research, (86), (1999). 53–62
[27] R. Nauss. Parametric Integer Programming, University of Missouri Press, Columbia, Missouri, 1979
[28] R. Ramaswamy. Sensitivity analysis in combinatorial optimization, Fellowship dissertation, Indian Institute of Management, Calcutta, India (1994)
[29] R. Ramaswamy and N. Chakravarti. Complexity of determining exact tolerances for min-sum and min-max combinatorial optimization problems, WPS-247/95, Working Paper Series, Indian Institute of Management, Calcutta, India (1995)
[30] K. Richter and J. Vörös. A parametric analysis of the dynamic lot-sizing problem. Journal of Information Processing and Cybernetics, 25(3), (1989). 67–73
[31] D.R. Shier and C. Witzgall. Arc tolerances in shortest path and network flow problems, Networks, 10(4), (1980). 277-291
[32] Y.N. Sotskov. Stability of high-speed optimal schedules, Computational Mathematics and Mathematical Physics, 29(3), (1989). 57–63
[33] Y.N. Sotskov. Stability of an optimal schedule, European Journal of Operational Research, 55, (1991). 91–102
[34] Y.N. Sotskov, V.K. Leontev and E.N. Gordeev. Some concepts of stability analysis in combinatorial optimization, Discrete Applied Mathematics, 58(2), (1995). 169–190
[35] Y.N. Sotskov, N.Y. Sotskova and F. Werner. Stability of an optimal schedule in a job shop, OMEGA, 25(4), (1997). 397–414
[36] Y.N. Sotskov, A.P.M. Wagelmans and F. Werner. On the calculation od the stability radius of an optimal or an approximate schedule, Annals of Operations Research, 83, (1998). 213–252
[37] P. M. Spira and A. Pan, On finding and updating spanning trees and shortest paths, SIAM Journal of Computing, 4, (1975) 375-380.
[38] R.E. Tarjan. Sensitivity analysis of minimum spanning trees and shortest path problems, Information Processing Letters, 14(1), (1982). 30–33
[39] E.S. van der Poort. Aspects of sensitivity analysis for the traveling salesman problem, Ph.D. thesis, Graduate school/Research Institute, Systems Organization and Management, University of Groningen, The Netherlands, (1997)
[40] E.S. van der Poort, V. Dimitrijević, G. Sierksma, and J.A.A. van der Veen. Using stability information for finding the k-best traveling salesman problem, Report

97A29, Graduate school/Research Institute, Systems Organization and Management, University of Groningen, The Netherlands, (1997)

[41] E.S. van der Poort, M. Libura, G. Sierksma, and J.A.A. van der Veen. Sensitivity analysis based on k-best solutions of the traveling salesman problem, Report 96A14, Graduate school/Research Institute, Systems Organization and Management, University of Groningen, The Netherlands, (1996)

[42] E.S. van der Poort, G. Sierksma, and J.A.A. van der Veen. Determining tolerances for the traveling salesman problem; a survey, Report 97A27, Graduate school/Research Institute, Systems Organization and Management, University of Groningen, The Netherlands, (1997) (To appear, Journal of Combinatorial Optimization)

[43] C.P.M. van Hoesel, A.W.J. Kolen, A.H.G. Rinnooy Kan and A.P.M. Wagelmans. Sensitivity analysis in combinatorial optimization: A bibliography, Technical Report 8944/A, Econometric Institute, Erasmus University, Rotterdam, The Netherlands, (1989)

[44] S. van Hoesel and A.P.M. Wagelmans. Sensitivity analysis of the economic lot-sizing problem, Discrete Applied Mathematics, 45(3), (1993). 291–312

[45] S. van Hoesel and A.P.M. Wagelmans. On the complexity of postoptimality analysis of 0/1 programs, Discrete Applied Mathematics, 91(1–3), (1999). 251–263

[46] A.P.M. Wagelmans. Sensitivity analysis in combinatorial optimization, Ph.D. Thesis, Econometric Institute, Erasmus University, Rotterdam, The Netherlands, (1990)

[47] C.J. Woeginger. Sensitivity analysis for knapsack problems: Another negative result, Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science, 92(2–3), (1999). 247–251